PROCESADORES DE LENGUAJE

Ingeniería Informática
Primer curso de segundo ciclo

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba
Curso académico 2014 – 2015

Hoja de ejercicios nº 2.- **FUNDAMENTOS TEÓRICOS DEL ANÁLISIS SINTÁCTICO**

Derivaciones y árboles sintácticos

1. Una gramática de contexto libre G posee el siguiente conjunto de producciones:

   P = { <oración> → <sujeto> <predicado>
   <sujeto> → <grupo_nominal>
   <sujeto> → <grupo_nominal> **ADJETIVO**
   <grupo_nominal> → **NOMBRE** | **ARTÍCULO NOMBRE**
   <predicado> → **VERBO** <complementos>
   <complementos> → <directo> <indirecto> <circunstanciales>
   <complementos> → <indirecto> <circunstanciales>
   <complementos> → <directo> <circunstanciales>
   <complementos> → <circunstanciales>
   <directo> → <grupo_nominal> | <grupo_nominal> **ADJETIVO**
   <indirecto> → **PREPOSICIÓN** <grupo_nominal>
   <circunstanciales> → ε
   <circunstanciales> → <circunstancial> <circunstanciales>
   <circunstancial> → **PREPOSICIÓN** <grupo_nominal>
   }

   a. Indica cuáles son los símbolos terminales y cuáles los no terminales
   b. ¿Cuál es el símbolo inicial?
   c. ¿Es la gramática recursiva por la izquierda o por la derecha?
   d. Muestra las derivaciones por la izquierda y por la derecha de la siguiente cadena:
      La niña entregó la llave mágica a un amigo en el bosque
   e. Dibuja el árbol sintáctico asociado a la derivación por la izquierda.

2. La siguiente gramática genera expresiones aritméticas con notación prefija:

   P = {
   E → **(** O L **)**
   O → **+** | **-** | ***** | **/**
   L → A | A L
   A → **IDENTIFICADOR** | **NÚMERO** | E
   }

   a. Indica cuáles son los símbolos terminales y cuáles los no terminales
   b. ¿Cuál es el símbolo inicial?
   c. ¿Es la gramática recursiva por la izquierda o por la derecha?
   d. Muestra las derivaciones por la izquierda y por la derecha de la siguiente expresión:
      (+ (* a a) (* b b))

e. Muestra los árboles sintácticos asociados a las derivaciones del apartado anterior y comprueba si son iguales.

3. La siguiente gramática genera sentencias del lenguaje Pascal:

P = { <asignación_lógica> → **IDENTIFICADOR :=** <predicado>
<predicado> → <predicado> **OR** <disyunción>
<predicado> → <disyunción>
<disyunción> → <disyuncion> **AND** <conjunción>
<disyunción> → <conjunción>
<conjunción> → <simple> | **NOT (** <predicado> **)**
<simple> → **(** <predicado> **)**
<simple> → <operando> <operador_relacional> <operando>
<operador_relacional> → **=** | **<** | **<=** | **>** | **>=** | **<>**
<simple> → **true** | **false**
<operando> → **IDENTIFICADOR** | **NÚMERO** | **true** | **false**
}

a. Indica cuáles son los símbolos terminales y cuáles los no terminales
b. ¿Cuál es el símbolo inicial?
c. ¿Es la gramática recursiva por la izquierda o por la derecha?
d. Muestra las derivaciones por la izquierda de las siguientes sentencias:
   - *estado := (final <> **true**)*
   - *apto := (not ((teoría < 4) **or** (prácticas < 4))) **and** (media >= 5)*
e. Dibuja el árbol sintáctico asociado a la derivaciones por la izquierda

Recomendación: renombra los símbolos no terminales

Diseño de gramáticas

4. Diseña gramáticas de contexto libre que generen los lenguajes que se indican:
   - $L_1 = \{ x \mid x = \mathbf{a}\ y\ \mathbf{b} \ \wedge \ y \in \{0,1\}^* \}$
   - $L_2 = \{ \mathbf{a}^{i}\, \mathbf{c}^{2j}\, \mathbf{b}^{i} \mid i, j > 0 \}$
   - $L_3 = \{ \mathbf{a}^{2i}\, \mathbf{b}^{i} \mid i > 0 \}$
   - $L_4 = \{ \mathbf{a}^{i}\, \mathbf{b}^{j}\, c^{k} \mid i, j, k > 0 \ \wedge j = i + k \}$
   - $L_5 = \{ x \mid x$ tiene igual número de ceros que de unos $\}$
   - $L_6 = \{ w\, w^{R} \mid w \in \{0,1\}^* \wedge w^{R}$ es la palabra inversa o refleja de w $\}$

5. Diseña gramáticas de contexto libre que permitan generar las siguientes sentencias del lenguaje de programación **C**:
   - Proposiciones lógicas, como por ejemplo:
     (a ==b) && (c != 0 || d >= 1)
   - Sentencias de control de C: **if, while, for** y **switch.**

6. Diseña gramáticas que permitan generar algunas de las declaraciones del lenguaje de programación Pascal:
   - Declaraciones de variables simples
     a, b, c: integer;
     d, e: integer := 9;
     x, y: real;
     z: real := 7.5;
   - Declaraciones de arrays
     vector_rango: array [-10 .. 10] of real;
     datos: array [7 .. 12] of integer := [90, 18, 23, -12, 37, 10];

Ambigüedad

7. Una gramática de contexto libre G posee el siguiente conjunto de producciones:
   - P = {S → **a** | S **a** | **b** S S | S S **b** | S **b** S}
     a. Comprueba que es ambigua generando dos derivaciones por la izquierda (o por la derecha) diferentes.
     b. Construye los árboles sintácticos asociados a esas derivaciones.

8. Una gramática de contexto libre G posee el siguiente conjunto de producciones:
   - P = { S → A | B
     A → **a** A **b** | **a b**
     B → **a b** B | ε
     }
     a. Indica el lenguaje que genera esta gramática.
     b. Comprueba que la gramática es ambigua y diseña otra equivalente que no lo sea.

9. Demuestra que si una gramática de contexto libre posee la siguiente característica entonces ha de ser ambigua:
   "Existe un símbolo no terminal "A" que posee, simultáneamente, alguna producción recursiva por la izquierda (A → A a) y alguna producción recursiva por la derecha (A→ b A)".

Operaciones de limpieza

10. Dadas las siguientes gramáticas, construye otras equivalentes sin símbolos inútiles.
    - P = {S → A C B d | B a B,
      A → a A d | B C a | a b,
      B → b B b | a,
      C → C A C | A C c}
    - P = {S → A B | C A d,
      A → a A b | b b A | a a,
      B → b A C | a B | B A,
      C → b a C a c | a b D,
      D → b D b c | c C a}

11. Dadas las siguientes gramáticas:
    - P = {S → L **IDENTIFICADOR :=** E ; , L → L **IDENTIFICADOR :=** | ε ,
      E → E + T | T , T → **IDENTIFICADOR** | **NÚMERO** }
    - P = {S → a A a | b B b | A B, A → a A a | ε, B → b B b | ε }
      a. Obtén otras gramáticas equivalentes sin reglas épsilon.
      b. Suprime las reglas unitarias de las gramáticas obtenidas en el apartado anterior.

Recursividad y factorización

12. Una gramática de contexto libre G posee el siguiente conjunto de producciones:
    P = {
    <expresión-relacional> → **(**<operador-relacional> <argumentos> **)**
    <operador-relacional> → **<** | **<=** |**=** | **>** | **>=**
    <argumentos> → <argumentos> <argumento> | <argumento>
    <argumento> → **NÚMERO** | **IDENTIFICADOR**
    }

a. Elimina la recursividad por la izquierda y factorízala por la izquierda.
b. Utiliza la gramática obtenida en el apartado anterior para generar la derivación por la izquierda y el árbol sintáctico de las siguiente sentencia:
(<= 0 temperatura 100)

- **Recomendación**: renombra los símbolos no terminales

13. La siguiente gramática permite generar asignaciones de expresiones aritméticas:

P = {
A → **IDENTIFICADOR =** E
E → T | E **+** T
T → P |T ***** P
P → F | F **^** P
F → **(** E **)** | **NÚMERO** | **IDENTIFICADOR**
}

a. Elimina la recursividad por la izquierda y factoriza la gramática por la izquierda.
b. Utiliza la gramática obtenida en el apartado anterior para generar la derivación por la izquierda y el árbol sintáctico de las siguiente sentencia:
h = (a^2 + b ^2) ^ 0.5

14. La siguiente gramática permite generar algunas de las **enumeraciones** del lenguaje C.

P = {
S → S E
S → E
E → **enum IDENTIFICADOR {** L **}** ;
L → L, I
L → I
I → **IDENTIFICADOR**
I → **IDENTIFICADOR = NÚMERO**
}

a. Elimina la recursividad por la izquierda y factoriza la gramática por la izquierda.
b. Utiliza la gramática obtenida en el apartado anterior para generar la derivación por la izquierda y el árbol sintáctico de las siguiente sentencia:
enum color { blanco, negro = -1, amarillo = 9, rojo };

Formas normales

15. Dada la siguiente gramática de contexto libre G:

P = { S → T L **;**
T → **INT** | **FLOAT**
L → **IDENTIFICADOR** | **IDENTIFICADOR** L'
L' → **,** **IDENTIFICADOR** | **,** **IDENTIFICADOR** L'
}

a. Obtén la forma normal de Chomsky
b. Obtén la forma normal de Greibach